ARMY RESEARCH LABORATORY

Creating, Positioning, and Rotating Rectangles Using C++

by Robert J. Yager

ARL-TN-558 August 2013

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

Army Research Laboratory
Aberdeen Proving Ground, MD 21005-5066

ARL-TN-558 **August 2013**

Creating, Positioning, and Rotating Rectangles Using C++

Robert J. Yager
Weapons and Materials Research Directorate, ARL

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE *(DD-MM-YYYY)*
August 2013

2. REPORT TYPE
Final

3. DATES COVERED (From - To)
March 2013

4. TITLE AND SUBTITLE
Creating, Positioning, and Rotating Rectangles Using C++

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Robert J. Yager

5d. PROJECT NUMBER
AH80

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
U.S. Army Research Laboratory
ATTN: RDRL-WML-A
Aberdeen Proving Ground, MD 21005-5066

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-TN-558

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This report documents a set of functions, written in C++, that can be used to create, position, and rotate rectangles, as well as test for intersection between rectangles. The functions build on functions from the y2DOps namespace (Yager, R. J. *Two-Dimensional Translations, Rotations, and Intersections Using C++*: U.S. Army Research Laboratory: Aberdeen Proving Ground, MD, to be submitted for publication). The functions have been grouped into the yRectangle2D namespace, which is summarized at the end of this report.

15. SUBJECT TERMS
rectangle, 2D, C++, translate, rotate, intersection, intersect

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
					Robert J. Yager
REPORT	**b. ABSTRACT**	**c. THIS PAGE**			**19b. TELEPHONE NUMBER** (*Include area code*)
Unclassified	Unclassified	Unclassified	UU	24	410-278-6689

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Contents

List of Figures **v**

Acknowledgments **vi**

1. Introduction **1**

2. Rectangles in Two Dimensions **1**

3. Creating Rectangles – the NewRectangle2D() Function **2**

3.1 NewRectangle2D() Code 2

3.2 NewRectangle2D() Parameters 2

3.3 NewRectangle2D() Return Value 3

3.4 NewRectangle2D() Example 3

4. Translating Rectangles – the TranslateRectangle2D() Function **3**

4.1 TranslateRectangle2D() Code 4

4.2 TranslateRectangle2D() Parameters 4

4.3 TranslateRectangle2D() Example 4

5. Rotating Rectangles – the RotateRectangle2D() Function **5**

5.1 RotateRectangle2D() Code 6

5.2 RotateRectangle2D() Parameters 6

5.3 RotateRectangle2D() Example 6

6. Checking for Rectangle Overlap – the RectangleIntersect2D() Function **7**

6.1 RectangleIntersect2D() Code 7

6.2 RectangleIntersect2D() Parameters 7

6.3 RectangleIntersect2D() Return Value 8

6.4 RectangleIntersect2D() Example 8

7. Interior/Exterior Check – the RectangleInterior2D() Function **9**

7.1 RectangleInterior2D() Code 9

7.2 RectangleInterior2D() Parameters....................9
7.2 RectangleInterior2D() Return Value....................9
7.2 RectangleInterior2D() Example....................10

8. Example – Placing *N* Uniformly Distributed Rectangles in a Rectangular Region 10

9. Summary 13

Distribution List 15

List of Figures

Figure 1. A rectangle with corners defined by position vectors $\vec{R}_0$, $\vec{R}_1$, $\vec{R}_2$, and $\vec{R}_3$......................1

Figure 2. A rectangle created using the NewRectangle2D() function. 2

Figure 3. A rectangle that has been repositioned using the TranslateRectangle2D() function........4

Figure 4. A rectangle that has been rotated using the RotateRectangle2D() function.....................5

Figure 5. Intersecting rectangles: A and B intersect, A and C do not. 7

Figure 6. Interior Points: $\vec{P}$ is interior to B, but not to A 9

Figure 7. *N* uniformly distributed rectangles. 12

Figure 8. Randomly generated points that are interior to 800 uniformly distributed rectangles. ..12

Acknowledgments

The author would like to thank Mr. Luke Strohm of the U.S. Army Research Laboratory's (ARL's) Weapons and Materials Research Directorate. Mr. Strohm provided technical and editorial recommendations that improved the quality of this report.

1. Introduction

This report documents a set of functions, written in C++, that can be used to create, position, and rotate rectangles, as well as test for intersection between rectangles. The functions build on functions from the y2DOps namespace.[1] The functions have been grouped into the yRectangle2D namespace, which is summarized at the end of this report.

2. Rectangles in Two Dimensions

Suppose that a rectangle R is defined by four position vectors $\vec{R}_0$, $\vec{R}_1$, $\vec{R}_2$, and $\vec{R}_3$ as shown in figure 1.

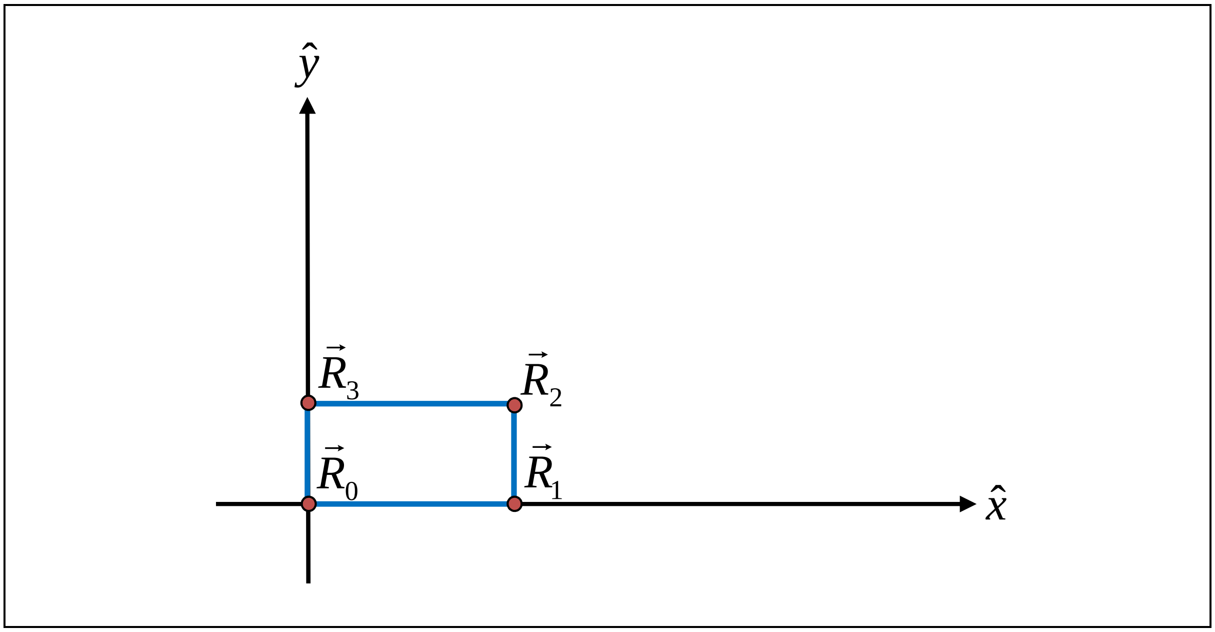


Figure 1. A rectangle with corners defined by position vectors $\vec{R}_0$, $\vec{R}_1$, $\vec{R}_2$, and $\vec{R}_3$.

[1] Yager, R. J. *Two-Dimensional Translations, Rotations, and Intersections Using C++*; TN 539; U.S. Army Research Laboratory: Aberdeen Proving Ground, MD, June 2013.

The functions described in this report operate on rectangles that are stored in eight-element arrays, where an array, **R**, is defined such that

$$\mathbf{R}=\{\,R_{0,x}\,,R_{0,y}\,,R_{1,x}\,,R_{1,y}\,,R_{2,x}\,,R_{2,y}\,,R_{3,x}\,,R_{3,y}\,\}. \tag{1}$$

3. Creating Rectangles – the NewRectangle2D() Function

The NewRectangle2D() function can be used to create rectangles that are placed with their lower-left corners at the origin, as shown in figure 2.

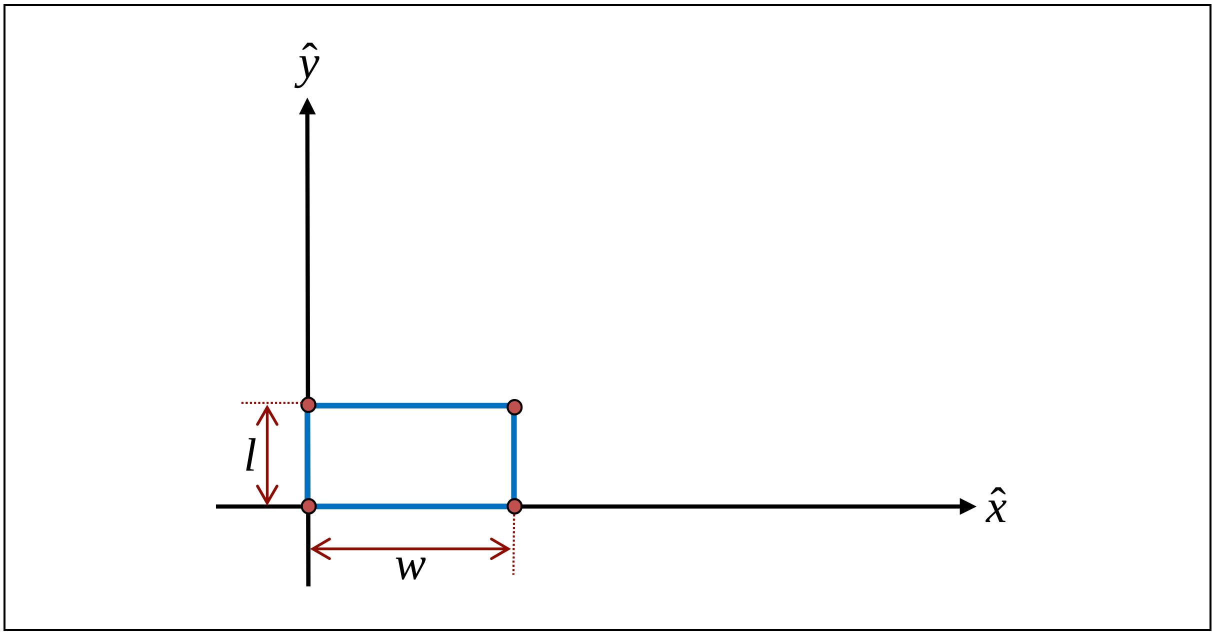


Figure 2. A rectangle created using the NewRectangle2D() function.

Note that the NewRectangle2D() function uses the “new” command to allocate memory for the array that is pointed to by its return value. Thus, to avoid memory leaks, each use of the NewRectangle2D() function should be accompanied by a use of the “delete[]” operator.

3.1 NewRectangle2D() Code

```
inline double* NewRectangle2D(//<======================CREATES A NEW RECTANGLE
    double w,//<--------------------------------------------WIDTH OF RECTANGLE
    double l){//<------------------------------------------LENGTH OF RECTANGLE
  double *R=new double[8];/*<-*/R[0]=R[1]=R[3]=R[6]=0,R[2]=R[4]=w,R[5]=R[7]=l;
  return R;
};//~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~10JUL2013~~~~~~
```

3.2 NewRectangle2D() Parameters

w **w** specifies the width of a rectangle.

l **l** specifies the length of a rectangle.

3.3 NewRectangle2D() Return Value

The NewRectangle2D() function returns a pointer to a new rectangle.

3.4 NewRectangle2D() Example

The following example begins by using the NewRectangle2D() function to create a rectangle that is 2.0 units wide and 1.0 unit long. The coordinates of the corners of the rectangle are then printed to the screen. Finally, the memory that was allocated using the NewRectangle2D() function is deallocated using the delete[] command.

```
#include <cstdio>//.....................................................printf()
#include "y_rectangle_2d.h"
int main(){
  double* R=yRectangle2D::NewRectangle2D(2,1);
  printf("   x  ,   y\n");
  for(int i=0;i<4;++i)printf(" %4.1f , %4.1f\n",R[2*i],R[2*i+1]);
  delete[] R;
};//~~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~10JUL2013~~~~~~
```

OUTPUT:

```
   x  ,   y
  0.0 ,  0.0
  2.0 ,  0.0
  2.0 ,  1.0
  0.0 ,  1.0
```

4. Translating Rectangles – the TranslateRectangle2D() Function

The TranslateRectangle2D() function can be used to reposition rectangles (without rotation) by a displacement vector $\vec{d}$, as shown in figure 3.

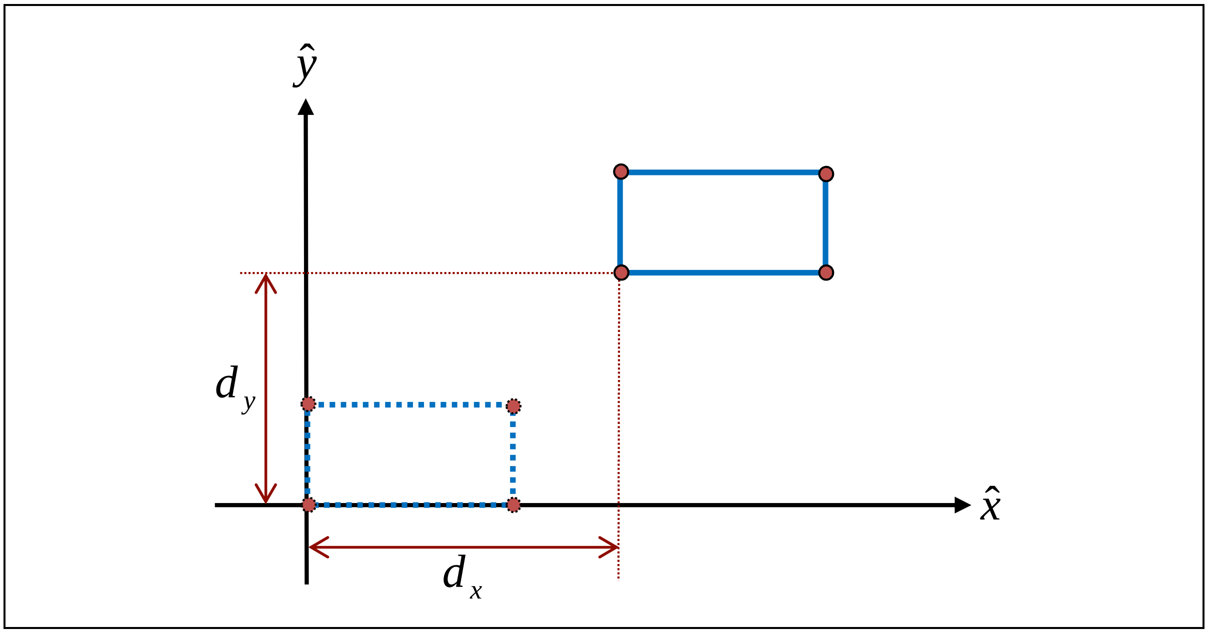


Figure 3. A rectangle that has been repositioned using the TranslateRectangle2D() function.

4.1 TranslateRectangle2D() Code

```
  inline void TranslateRectangle2D(//<====================TRANSLATES A RECTANGLE
      double R[8],//<----------------A RECTANGLE (CREATE USING NewRectangle2D())
      const double d[2]){//<---------------------------------DISPLACEMENT VECTOR
    for(int i=0;i<4;++i)y2DOps::Translate2D(R+2*i,d);
  };//~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~10JUL2013~~~~~~
```

4.2 TranslateRectangle2D() Parameters

R **R** specifies a rectangle that has been created using the NewRectangle2D() function.

d **d** specifies a two-element array that stores the displacement vector $\vec{d}$ (**d**=$\{d_x, d_y\}$). **d** determines the magnitude and direction by which **R** is translated.

4.3 TranslateRectangle2D() Example

The following example begins by using the NewRectangle2D() function to create a rectangle that is 2.0 units wide and 1.0 unit long. The rectangle is then translated by 3.0 units in the x direction and 2.5 units in the y direction using the TranslateRectangle2D() function. The coordinates of the corners of the translated rectangle are then printed to the screen. Finally, the memory that was allocated using the NewRectangle2D() function is deallocated using the delete[] command.

```
#include <cstdio>//.....................................................printf()
#include "y_rectangle_2d.h"
int main(){
  double* R=yRectangle2D::NewRectangle2D(2,1);
  double d[2]={3,2.5};
  yRectangle2D::TranslateRectangle2D(R,d);
  printf("   x  ,   y\n");
  for(int i=0;i<4;++i)printf(" %4.1f , %4.1f\n",R[2*i],R[2*i+1]);
  delete[] R;
};//~~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~10JUL2013~~~~~~
```

OUTPUT:

```
   x  ,   y
  3.0 ,  2.5
  5.0 ,  2.5
  5.0 ,  3.5
  3.0 ,  3.5
```

5. Rotating Rectangles – the RotateRectangle2D() Function

The RotateRectangle2D() function can be used to rotate rectangles about their centers by an angle, θ, as shown in figure 4.

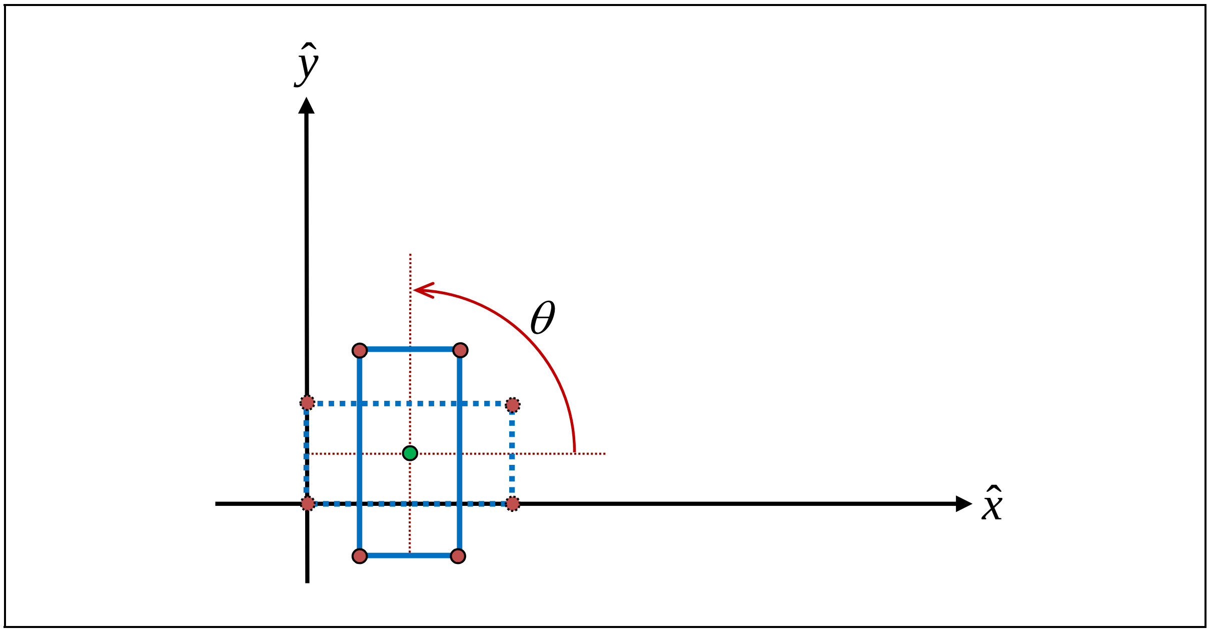


Figure 4. A rectangle that has been rotated using the RotateRectangle2D() function.

5.1 RotateRectangle2D() Code

```
  inline void RotateRectangle2D(//<=========ROTATES A RECTANGLE ABOUT ITS CENTER
      double R[8],//<----------------A RECTANGLE (CREATE USING NewRectangle2D())
      double rads){//<---------------THE ANGLE OF THE ROTATION (CCW IS POSITIVE)
    double M[4],o[2]={(R[0]+R[2]+R[4]+R[6])/4,(R[1]+R[3]+R[5]+R[7])/4};
    y2DOps::RMatrix2D(M,rads);
    for(int j=0;j<4;++j)y2DOps::Rotate2D(R+2*j,o,M);
  };//~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~10JUL2013~~~~~~
```

5.2 RotateRectangle2D() Parameters

R **R** specifies a rectangle that has been created using the NewRectangle2D() function.

rads **rads** specifies the angle (in radians) of a rotation. The direction of the rotation is counterclockwise.

5.3 RotateRectangle2D() Example

The following example begins by using the NewRectangle2D() function to create a rectangle that is 2.0 units wide and 1.0 unit long. The rectangle is then rotated by $\pi/2$ using the RotateRectangle2D() function. The coordinates of the corners of the rotated rectangle are then printed to the screen. Finally, the memory that was allocated using the NewRectangle2D() function is deallocated using the delete[] command.

```
#include <cstdio>//.....................................................printf()
#include "y_rectangle_2d.h"
int main(){
  double w=2,l=1;//.............................................size of rectangle
  double* R=yRectangle2D::NewRectangle2D(w,l);
  yRectangle2D::RotateRectangle2D(R,3.14159265358979/2);
  printf("   x  ,   y\n");
  for(int i=0;i<4;++i)printf(" %4.1f , %4.1f\n",R[2*i],R[2*i+1]);
  delete[] R;
};//~~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~10JUL2013~~~~~~
```

OUTPUT:

```
   x  ,   y
  1.5 , -0.5
  1.5 ,  1.5
  0.5 ,  1.5
  0.5 , -0.5
```

6. Checking for Rectangle Overlap – the RectangleIntersect2D() Function

The RectangleIntersect2D() function can be used to determine whether or not any of the sides of two rectangles intersect. Figure 5 shows examples of intersecting and nonintersecting rectangles.

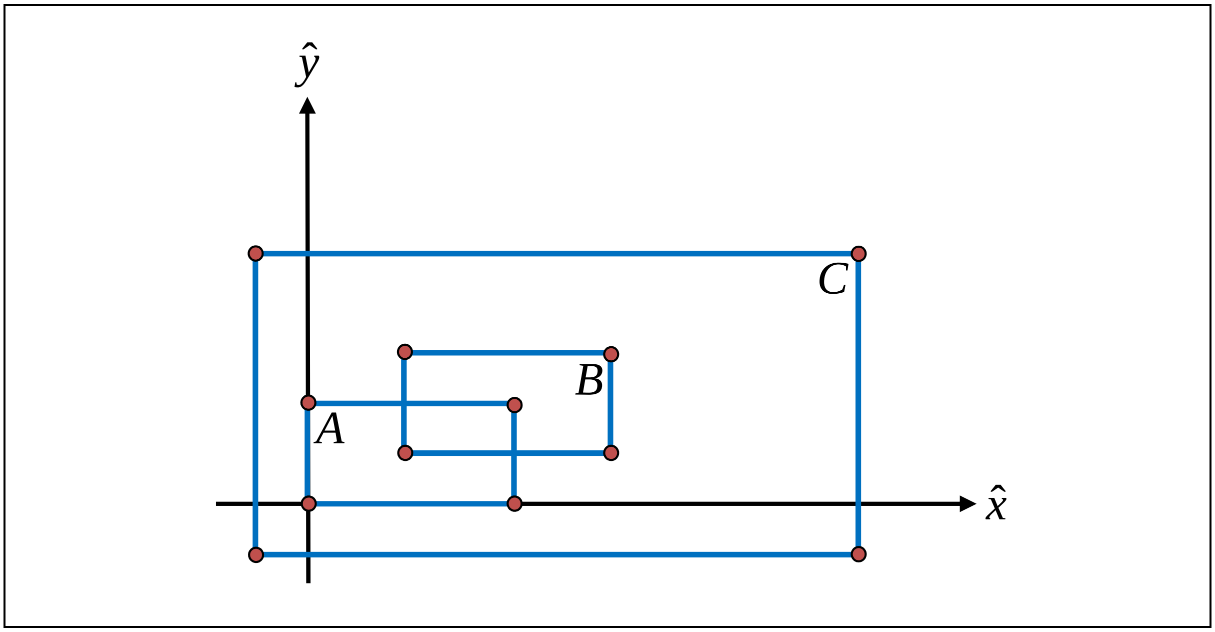


Figure 5. Intersecting rectangles: A and B intersect, A and C do not.

6.1 RectangleIntersect2D() Code

```
inline bool RectangleIntersect2D(//<===DO THE SIDES OF 2 RECTANGLES INTERSECT?
    const double R1[8],//<--------RECTANGLE #1 (CREATE USING NewRectangle2D())
    const double R2[8],//<--------RECTANGLE #2 (CREATE USING NewRectangle2D())
    double e=1E-9){//<--CUTOFF VALUE FOR DETERMINING IF TWO SIDES ARE PARALLEL
  for(int j=0;j<4;++j)for(int j2=0;j2<4;++j2){
    double t[2],x[2];
    double LA[4]={R1[2*j ],R1[2*j +1] , R1[(j +1)%4*2],R1[(j +1)%4*2+1]};
    double LB[4]={R2[2*j2],R2[2*j2+1] , R2[(j2+1)%4*2],R2[(j2+1)%4*2+1]};
    if(!y2DOps::IParameters2D(t,LA,LB))continue;
    if(y2DOps::Intersect2D(x,t,LA))return true;}
  return false;
};//~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~10JUL2013~~~~~~
```

6.2 RectangleIntersect2D() Parameters

R1 **R1** specifies a rectangle that has been created using the NewRectangle2D() function.

R2 **R2** specifies a second rectangle that has been created using the NewRectangle2D() function.

e **e** specifies the cutoff value for determining if two sides are parallel.

6.3 RectangleIntersect2D() Return Value

The RectangleIntersect2D() function returns true if any side of R1 intersects any side of R2, except for the special case where two sides intersect at more than one point.

6.4 RectangleIntersect2D() Example

The following example begins by creating and positioning the three rectangles shown in figure 5. The RectangleIntersect2D() function is then used to determine whether or not the rectangles intersect. The results are printed to the screen. Finally, the memory that was allocated using the NewRectangle2D() function is deallocated using the delete[] command.

```
#include <cstdio>//.....................................................printf()
#include "y_rectangle_2d.h"
int main(){
  double* A=yRectangle2D::NewRectangle2D(2,1);
  double* B=yRectangle2D::NewRectangle2D(2,1);
  double* C=yRectangle2D::NewRectangle2D(6,3);
  double d[2]={1,.5};
  yRectangle2D::TranslateRectangle2D(B,d);
  d[0]=-.5,d[1]=-.5;
  yRectangle2D::TranslateRectangle2D(C,d);
  printf("A intersects B? %s\n",
    yRectangle2D::RectangleIntersect2D(A,B)?"true":"false");
  printf("B intersects C? %s\n",
    yRectangle2D::RectangleIntersect2D(B,C)?"true":"false");
  printf("C intersects A? %s\n",
    yRectangle2D::RectangleIntersect2D(C,A)?"true":"false");
  delete[] A;
  delete[] B;
  delete[] C;
};//~~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~10JUL2013~~~~~~
```

OUTPUT:

```
A intersects B? true
B intersects C? false
C intersects A? false
```

7. Interior/Exterior Check – the RectangleInterior2D() Function

The RectangleInterior2D() function can be used to determine whether or not a point lies inside a rectangle.

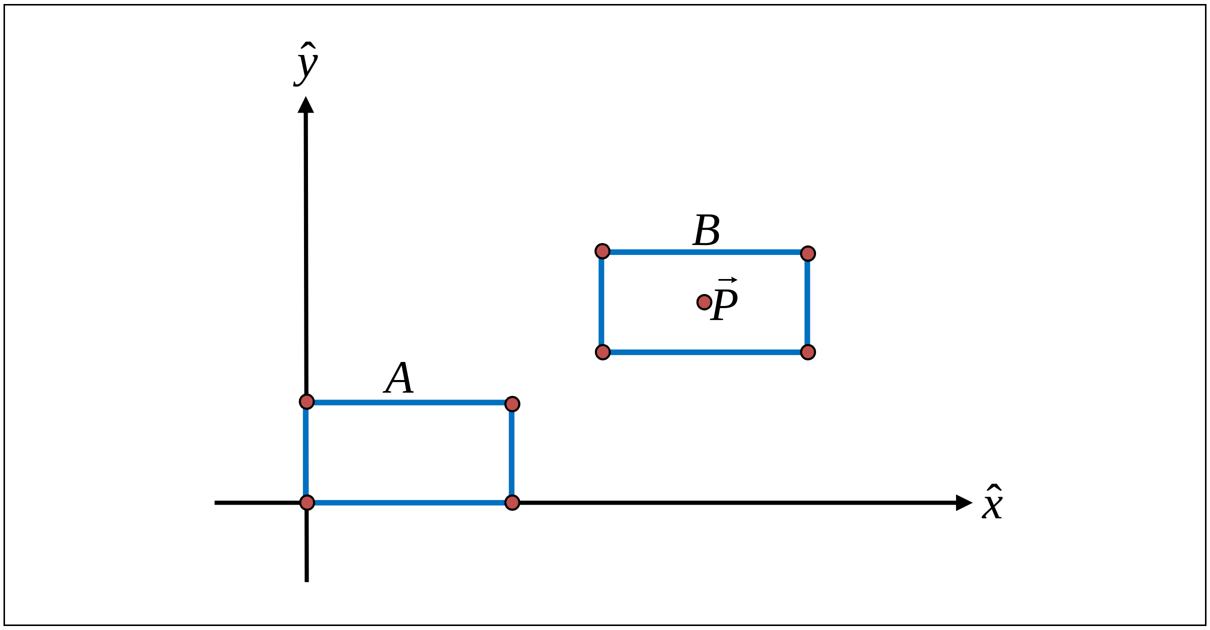


Figure 6. Interior Points: $\vec{P}$ is interior to B, but not to A.

7.1 RectangleInterior2D() Code

```
inline bool RectangleInterior2D(//<=============IS A POINT INSIDE A RECTANGLE?
    const double R[8],//<----------A RECTANGLE (CREATE USING NewRectangle2D())
    const double P[2]){//<---------------------------------------------A POINT
  double A[4]={P[0],P[1],P[0]+R[6]-R[0],P[1]+R[7]-R[1]};
  double B[4]={P[0],P[1],P[0]+R[2]-R[0],P[1]+R[3]-R[1]};
  double C[4]={R[2],R[3],R[4],R[5]};
  double D[4]={R[6],R[7],R[4],R[5]};
  double t1[2];/*<-*/y2DOps::IParameters2D(t1,A,D);
  double t2[2];/*<-*/y2DOps::IParameters2D(t2,B,C);
  return t1[0]>0&&t1[0]<1&&t2[0]>0&&t2[0]<1;
};//~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~10JUL2013~~~~~~
```

7.2 RectangleInterior2D() Parameters

R **R** specifies a rectangle that has been created using the NewRectangle2D() function.

P **P** specifies a point that lies in the plane of R.

7.2 RectangleInterior2D() Return Value

The RectangleInterior2D() function returns true if **P** lies interior to **R**, and false otherwise.

7.2 RectangleInterior2D() Example

The following example begins by creating and positioning the rectangles and the point that are shown in figure 6. The RectangleInterior2D() function is then used to determine whether or not the point is interior to either of the rectangles. The results are printed to the screen. Finally, the memory that was allocated using the NewRectangle2D() function is deallocated using the delete[] command.

```
#include <cstdio>//.....................................................printf()
#include "y_rectangle_2d.h"
int main(){
  double* A=yRectangle2D::NewRectangle2D(2,1);
  double* B=yRectangle2D::NewRectangle2D(2,1);
  double d[2]={3,2};
  yRectangle2D::TranslateRectangle2D(B,d);
  double P[2]={4,2.5};
  printf("P interior to A? %s\n",
    yRectangle2D::RectangleInterior2D(A,P)?"true":"false");
  printf("P interior to B? %s\n",
    yRectangle2D::RectangleInterior2D(B,P)?"true":"false");
  delete[] A;
  delete[] B;
};//~~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~10JUL2013~~~~~~
```

OUTPUT:

```
P interior to A? false
P interior to B? true
```

8. Example – Placing *N* Uniformly Distributed Rectangles in a Rectangular Region

The following example uses functions from the yRectangle2D namespace to place 800 rectangles that are 2.0 units by 3.0 units in size within a 200.0 unit by 100.0 unit rectangle. Figure 6 provides a visual of the 800 rectangles. Note that none of the rectangles intersect, either with each other, or with the bordering rectangle. The example then creates 1,000,000 randomly positioned points and tests to see if any of the points are interior to any of the rectangles. Figure 7 provides a visual of the points that are interior to rectangles.

```
#include <cstdio>//......................FILE,freopen(),stdout,fclose(),printf()
#include <cstdlib>//.............................................rand(),RAND_MAX
#include "y_rectangle_2d.h"
int main(){
  int N=800;//....................................number of rectangles in region
  double w=2,l=3;//...........................................size of rectangles
  double W=200,L=100;//...........................................size of region
  double* B=yRectangle2D::NewRectangle2D(W,L);//...................region border
  double** R=new double*[N];//...................storage for a set of rectangles
  FILE *f=freopen("rectangles.txt","w",stdout);//......redirect output to a file
  for(int i=0;i<N;++i){
    R[i]=yRectangle2D::NewRectangle2D(w,l);
    yRectangle2D::RotateRectangle2D(R[i],2*3.14159265358979*rand()/RAND_MAX);
    redo://.................if two rectangles intersect, come back to this point
    double d[2]={-w/2+W*rand()/RAND_MAX,-l/2+L*rand()/RAND_MAX};
    yRectangle2D::TranslateRectangle2D(R[i],d);
    if(yRectangle2D::RectangleIntersect2D(R[i],B)){
      d[0]*=-1,d[1]*=-1,yRectangle2D::TranslateRectangle2D(R[i],d);
      goto redo;}
    for(int i2=0;i2<i;++i2)if(yRectangle2D::RectangleIntersect2D(R[i],R[i2])){
      d[0]*=-1,d[1]*=-1,yRectangle2D::TranslateRectangle2D(R[i],d);
      goto redo;}
    printf("%f,%f,%f,%f,%f,%f,%f,%f\n",R[i][0],R[i][1],R[i][2],R[i][3],
                                       R[i][4],R[i][5],R[i][6],R[i][7]);}
  fclose(f);
  freopen("points.txt","w",stdout);//..................redirect output to a file
  for(int k=0;k<1000000;++k){
    double P[2]={W*rand()/RAND_MAX,L*rand()/RAND_MAX};
    for(int i=0;i<N;++i)if(yRectangle2D::RectangleInterior2D(R[i],P)){
      printf("%f,%f\n",P[0],P[1]);
      break;}}
  for(int i=0;i<N;++i)delete[] R[i];
  fclose(f);
};//~~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~DRAFT~~~~~~~~~~LAST~UPDATED~10JUL2013~~~~~~~
```

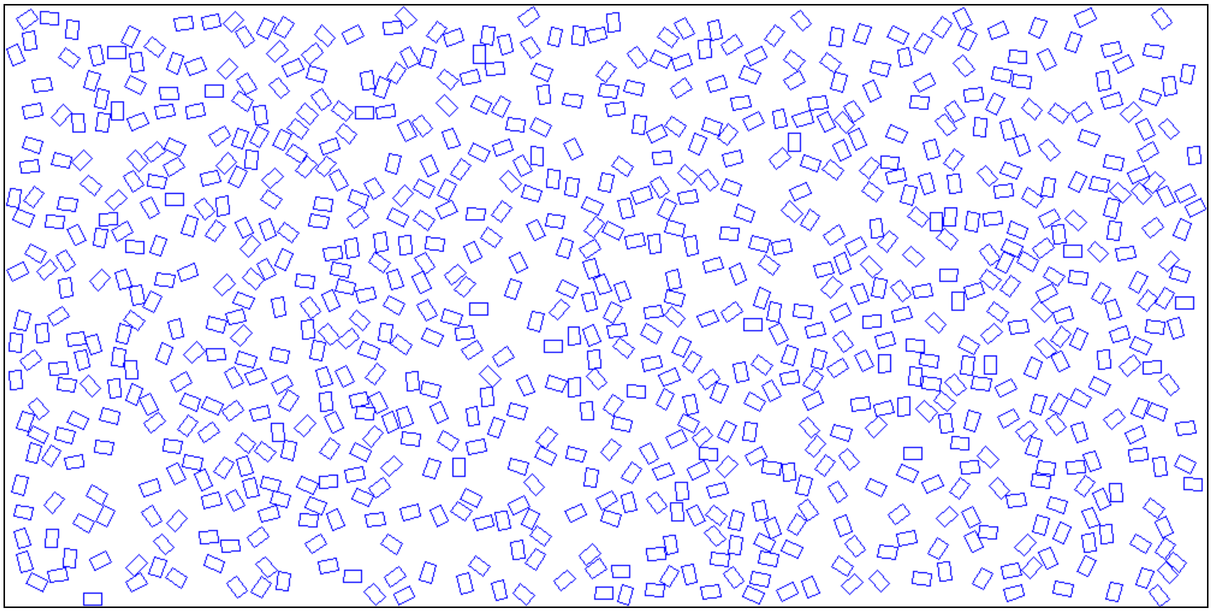


Figure 7. *N* uniformly distributed rectangles.

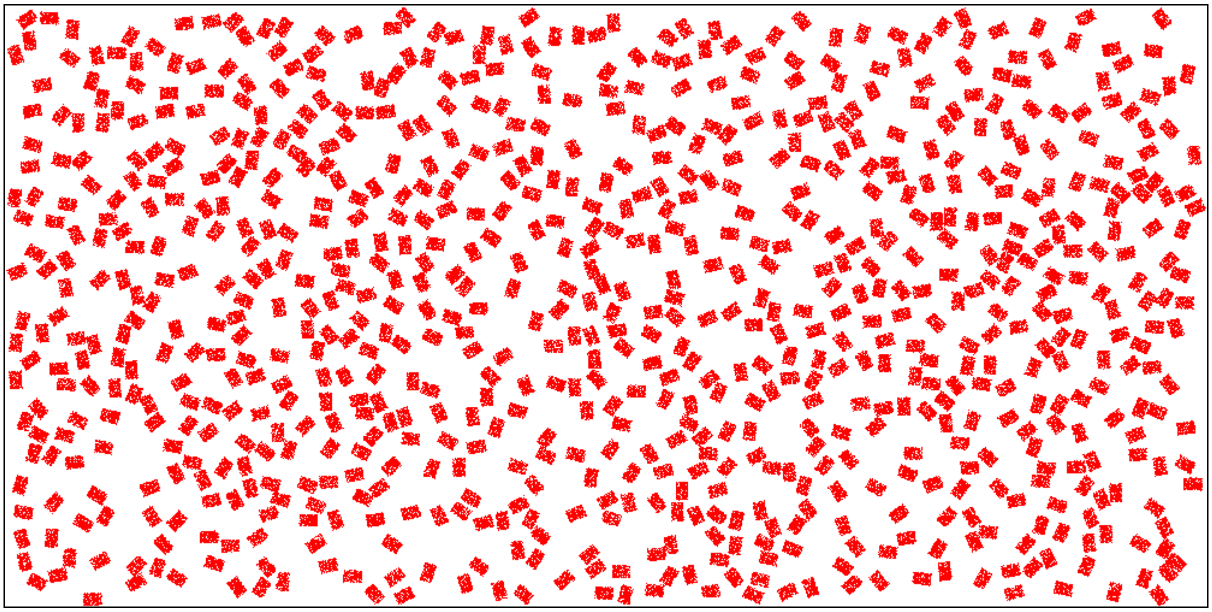


Figure 8. Randomly generated points that are interior to 800 uniformly distributed rectangles.

.

9. Summary

A summary sheet is provided at the end of this report. It presents the yRectangle2D namespace, which contains the five functions that are described in this report. Also presented is the y2DOps namespace, which is utilized by functions in the yRectangle2D namespace, and the example that is presented in section 8.

y_ rectangle_2d.h

```
#ifndef Y_RECTANGLE_2D_GUARD
#define Y_RECTANGLE_2D_GUARD
#include "y_2d_ops.h"
namespace yRectangle2D{
  //@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
  inline double* NewRectangle2D(//<======================CREATES A NEW RECTANGLE
      double w,//<--------------------------------------------WIDTH OF RECTANGLE
      double l){//<-----------------------------------------LENGTH OF RECTANGLE
    double *R=new double[8];/*<-*/R[0]=R[1]=R[3]=R[6]=0,R[2]=R[4]=w,R[5]=R[7]=l;
    return R;
  };//~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~10JUL2013~~~~~~
  inline void TranslateRectangle2D(//<====================TRANSLATES A RECTANGLE
      double R[8],//<----------------A RECTANGLE (CREATE USING NewRectangle2D())
      const double d[2]){//<---------------------------------DISPLACEMENT VECTOR
    for(int i=0;i<4;++i)y2DOps::Translate2D(R+2*i,d);
  };//~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~10JUL2013~~~~~~
  inline void RotateRectangle2D(//<=========ROTATES A RECTANGLE ABOUT ITS CENTER
      double R[8],//<----------------A RECTANGLE (CREATE USING NewRectangle2D())
      double rads){//<---------------THE ANGLE OF THE ROTATION (CCW IS POSITIVE)
    double M[4],o[2]={(R[0]+R[2]+R[4]+R[6])/4,(R[1]+R[3]+R[5]+R[7])/4};
    y2DOps::RMatrix2D(M,rads);
    for(int j=0;j<4;++j)y2DOps::Rotate2D(R+2*j,o,M);
  };//~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~10JUL2013~~~~~~
  inline bool RectangleIntersect2D(//<===DO THE SIDES OF 2 RECTANGLES INTERSECT?
      const double R1[8],//<--------RECTANGLE #1 (CREATE USING NewRectangle2D())
      const double R2[8],//<--------RECTANGLE #2 (CREATE USING NewRectangle2D())
      double e=1E-9){//<--CUTOFF VALUE FOR DETERMINING IF TWO SIDES ARE PARALLEL
    for(int j=0;j<4;++j)for(int j2=0;j2<4;++j2){
      double t[2],x[2];
      double LA[4]={R1[2*j ],R1[2*j +1] , R1[(j +1)%4*2],R1[(j +1)%4*2+1]};
      double LB[4]={R2[2*j2],R2[2*j2+1] , R2[(j2+1)%4*2],R2[(j2+1)%4*2+1]};
      if(!y2DOps::IParameters2D(t,LA,LB))continue;
      if(y2DOps::Intersect2D(x,t,LA))return true;}
    return false;
  };//~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~10JUL2013~~~~~~
  inline bool RectangleInterior2D(//<=============IS A POINT INSIDE A RECTANGLE?
      const double R[8],//<----------A RECTANGLE (CREATE USING NewRectangle2D())
      const double P[2]){//<---------------------------------------------A POINT
    double A[4]={P[0],P[1],P[0]+R[6]-R[0],P[1]+R[7]-R[1]};
    double B[4]={P[0],P[1],P[0]+R[2]-R[0],P[1]+R[3]-R[1]};
    double C[4]={R[2],R[3],R[4],R[5]};
    double D[4]={R[6],R[7],R[4],R[5]};
    double t1[2];/*<-*/y2DOps::IParameters2D(t1,A,D);
    double t2[2];/*<-*/y2DOps::IParameters2D(t2,B,C);
    return t1[0]>0&&t1[0]<1&&t2[0]>0&&t2[0]<1;
  };//~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~10JUL2013~~~~~~
}//@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
#endif
```

image created from rectangles.txt

(rotated 90 degrees)

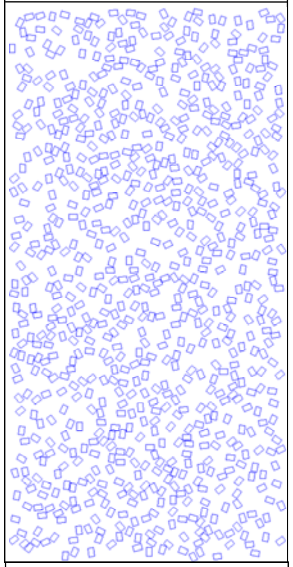
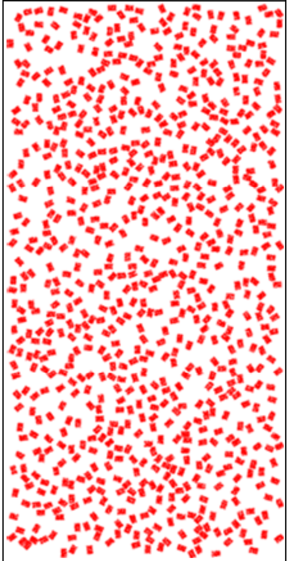


image created from points.txt

(rotated 90 degrees)



EXAMPLE

```
#define main yRectangle2DRectangularBattlefieldExample//@@@@@@@@@@@@@@@@@@@@@@@@
#include <cstdio>//......................FILE,freopen(),stdout,fclose(),printf()
#include <cstdlib>//.............................................rand(),RAND_MAX
#include "y_rectangle_2d.h"
int main(){
  int N=800;//....................................number of rectangles in region
  double w=2,l=3;//...........................................size of rectangles
  double W=200,L=100;//...........................................size of region
  double* B=yRectangle2D::NewRectangle2D(W,L);//...................region border
  double** R=new double*[N];//...................storage for a set of rectangles
  FILE *f=freopen("rectangles.txt","w",stdout);//......redirect output to a file
  for(int i=0;i<N;++i){
    R[i]=yRectangle2D::NewRectangle2D(w,l);
    yRectangle2D::RotateRectangle2D(R[i],2*3.14159265358979*rand()/RAND_MAX);
    redo://.................if two rectangles intersect, come back to this point
    double d[2]={-w/2+W*rand()/RAND_MAX,-l/2+L*rand()/RAND_MAX};
    yRectangle2D::TranslateRectangle2D(R[i],d);
    if(yRectangle2D::RectangleIntersect2D(R[i],B)){
      d[0]*=-1,d[1]*=-1,yRectangle2D::TranslateRectangle2D(R[i],d);
      goto redo;}
    for(int i2=0;i2<i;++i2)if(yRectangle2D::RectangleIntersect2D(R[i],R[i2])){
      d[0]*=-1,d[1]*=-1,yRectangle2D::TranslateRectangle2D(R[i],d);
      goto redo;}
    printf("%f,%f,%f,%f,%f,%f,%f,%f\n",R[i][0],R[i][1],R[i][2],R[i][3],
                                       R[i][4],R[i][5],R[i][6],R[i][7]);}
  fclose(f);
  freopen("points.txt","w",stdout);//..................redirect output to a file
  for(int k=0;k<1000000;++k){
    double P[2]={W*rand()/RAND_MAX,L*rand()/RAND_MAX};
    for(int i=0;i<N;++i)if(yRectangle2D::RectangleInterior2D(R[i],P)){
      printf("%f,%f\n",P[0],P[1]);
      break;}}
  for(int i=0;i<N;++i)delete[] R[i];
  fclose(f);
};//~~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~DRAFT~~~~~~~~~~LAST~UPDATED~10JUL2013~~~~~~
```

NO. OF COPIES	ORGANIZATION
1 (PDF)	DEFENSE TECHNICAL INFORMATION CTR DTIC OCA
1 (PDF)	DIRECTOR US ARMY RESEARCH LAB IMAL HRA
1 (PDF)	DIRECTOR US ARMY RESEARCH LAB RDRL CIO LL
1 (PDF)	GOVT PRINTG OFC A MALHOTRA
	ABERDEEN PROVING GROUND
1 (PDF)	DIR USARL RDRL WML A R YAGER

INTENTIONALLY LEFT BLANK.